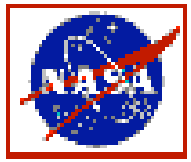
Technical Managers Training (TMT)
October 2005



Software Overview

Presented by: Barbara Pfarr

Barbara.B.Pfarr@nasa.gov

Associate Chief, Information Systems Division

Goddard Space Flight Center

Modified from Presentation by Mike Stark, "Mission Software for Project Managers"

Goals

- Describe the projects and activities of the Information Systems Division
- Describe best practices for software development
- Provide understanding of software issues

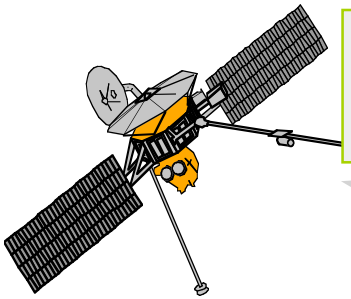
ISD: End-to-End Information Systems Providers

Joe Hennessy - Chief, Martha Chu, Barbara Pfarr- Associates, Julie Loftis – Assistant for Technology

Branch	Functional Area/Products	Services
581/Systems Integration & Engineering *Margaret Caulfield, vacant*	End-to-end data systems engineering of ISC mission systems development activities.	Mission directors, ground sys/flight ops management, sys. eng., flight prep support, SW eng, Sys I&T, AO prep
582/Flight Software *Elaine Shell, Ray Whitley, Kequan Luu, Yvonne. Lue*	Embedded spacecraft, instrument and hardware component softwares; FSW testbeds	End-to-end FSW development; simulation s/w; spacecraft sustaining engineering
583/Mission Applications *Henry Murray, Scott Green*	Off-line mission data systems (e.g., Command man., s/c mission and science P&S, GN&C, NCC	Sys. eng.& implementation, COTS application, testbeds for concept proof/prototyping in ops environment
584/Real-Time Software Engineering *John Donohue, Ryan Turner*	Real-time ground mission data systems for I&T and on-orbit ops (e.g., s/c command & control, launch and tracking services)	Sys. eng.& implementation, COTS application, simulators, testbeds for concept proof/prototyping in ops env.
585/Computing Environments & Technology *Howard Eiserike, Steve Naus*	Tools and services in support of information management	Network manage., business/support tool develop, WWW applications
586/Science Data Systems *Tom Flately, Bob Lutz*	Science data systems including data processing, archival, distribution, analysis & info man.	Sys. eng.& implementation, COTS application & integration, testbeds, prototyping
587/Advanced Data Management and Analysis *Jim Byrnes*	Advanced concept development for archival, retrieval, display, dissemination of science data	Next-gen req. development, testbed for sys evaluation, prototype products
588/Advanced Architectures & Autonomy *Vacant, Barbie Brown-Medina*	Technology R&D focused on space-ground automation and autonomy sys, advanced architectures, and advanced scientific tools and systems	Sys. eng & implementation, human-computer eng., technology evaluations, concept prototypes, sw eng. methods
589/Wallops System Software Engineering *Pam Pittman*	Mission application prototypes, including development I&T, sys engr., software engr. supporting BPO, RMMO OSB	Provide unique mission software applications, and project formulation services

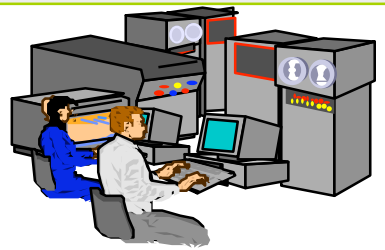
Mission Software: Architecture

End-to-end data systems engineering of mission systems

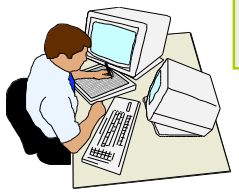


Embedded spacecraft, instrument and hardware component software

Real-time ground mission data systems for spacecraft integration and on-orbit ops (e.g., S/C command & control, launch and tracking services)



Science data systems including data processing, archival, distribution, analysis & info mgmt.



Off-line mission data systems (e.g., command mgmt., S/C mission and science planning & scheduling, guidance & navigation, network scheduling)

Different Domains of Software Each Reflect A Different Emphasis

Flight Software

- driven by limited S/C life, asset survival, & mission science program
- continuous critical real-time ops, e.g. attitude control, H&S monitoring
- fixed & constrained environment
- minimize risk with a never fail mindset
- constrained maintenance opportunities

Science Data Management & Data Processing

- data retention & integrity driven
- near-time and later ops, from raw archival to signature calibrations and analysis
- flexible & extendable environment
- data fail soft mindset
- - shadow mode and add-on maintenance

Mission Control Ground Systems

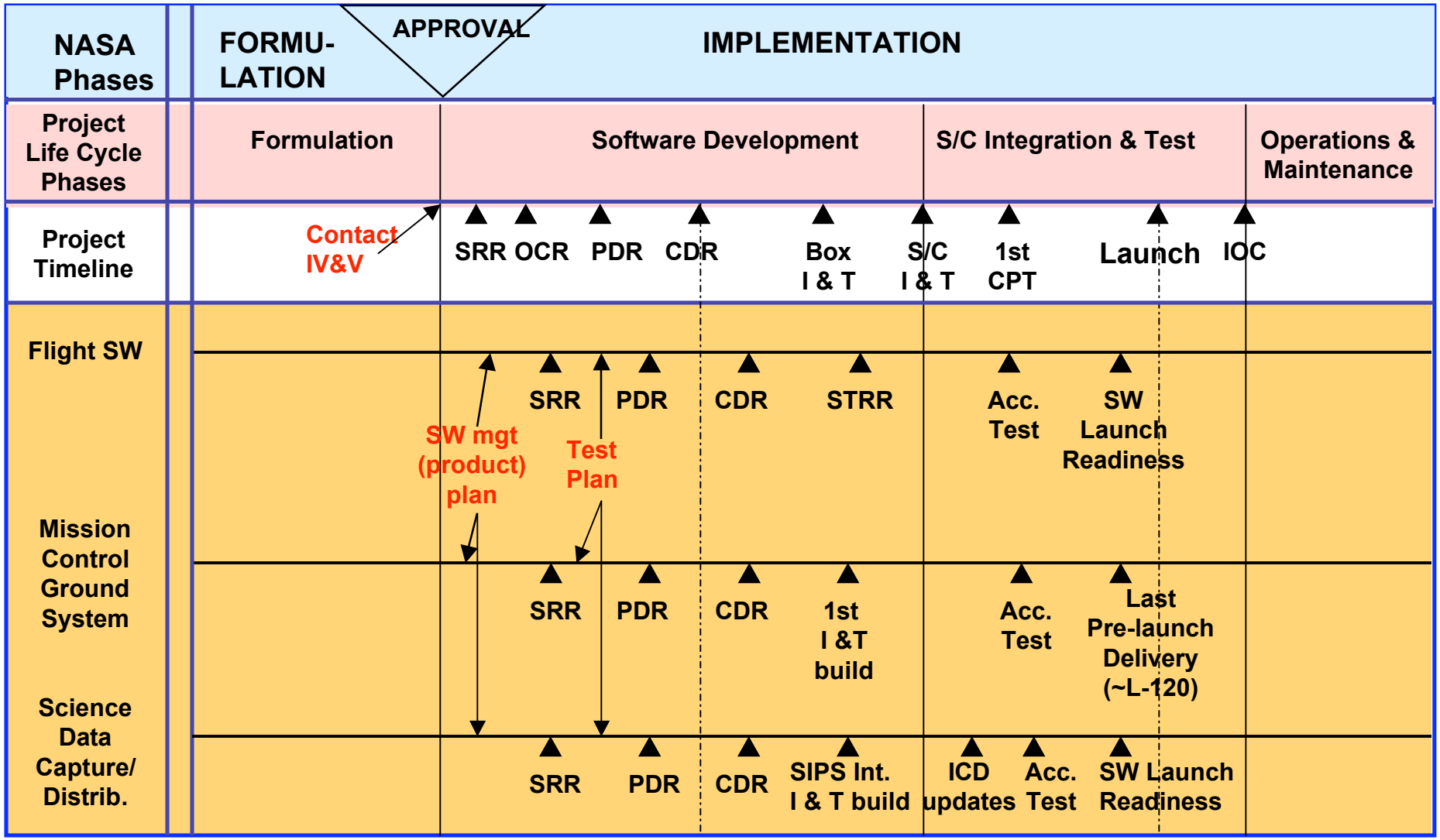
- driven by limited S/C life, asset health, and observatory user demands
- episodic real-time & near-time ops, from command uplink to system state evaluations
- open to needs based augmentation
- risk adverse with a fail soft/over mindset
- full shadow maintenance capability

Science Data Dissemination

- science evolution & user driven
- near-time and later ops
- large user communities
- evolving user interfaces & access demands
- timely data delivery mindset
- shadow mode and add-on maintenance

… Domain Tailored Development & Qualification Approaches

GSFC Project Life-cycle



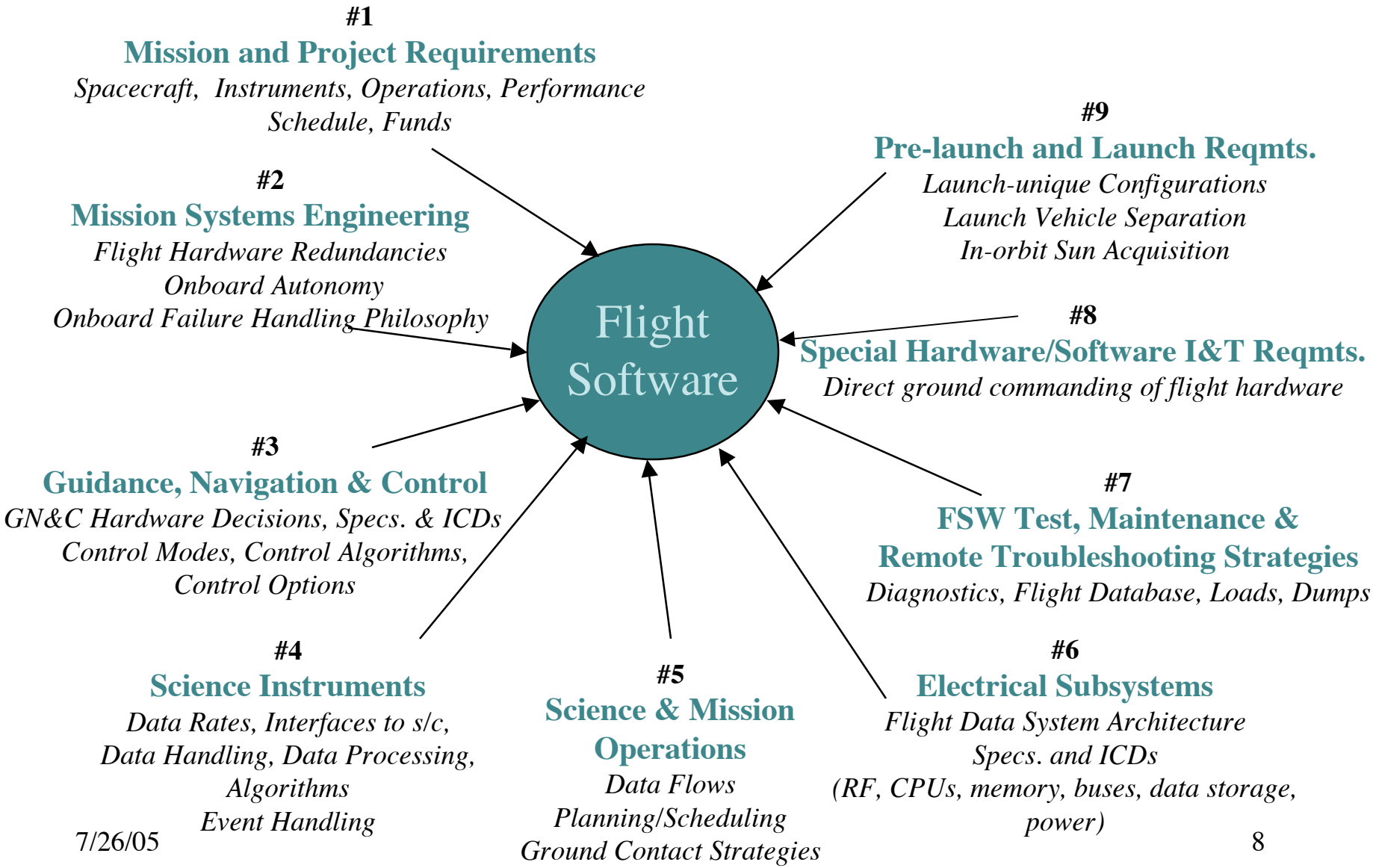
Agenda: Formulation Phase

- Key Documents and Deliverables
- Mission Software Architecture & Requirements
- Acquisition
- Cost estimation
- Software related trades

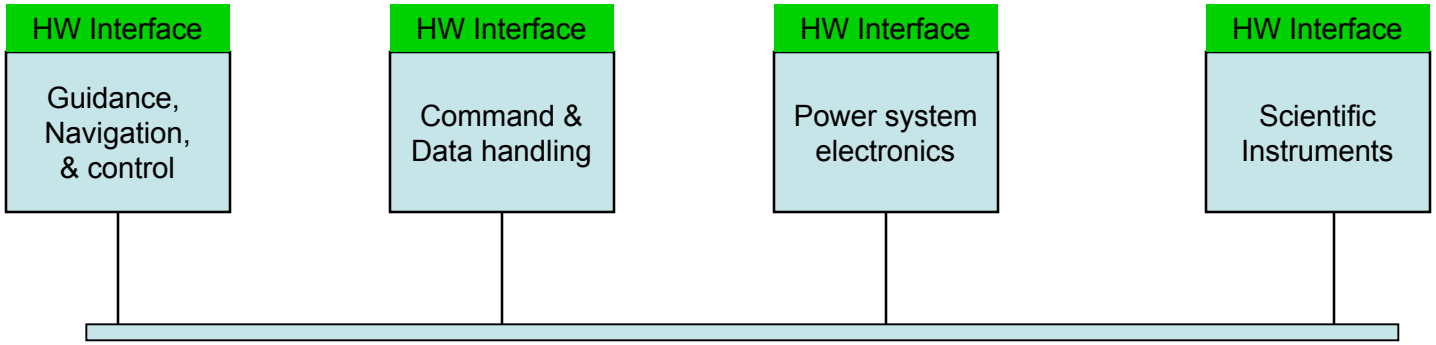
Key Documents / Deliverables

- Cost & Schedule Estimates
  - Input to SW management plan/product plan
- Flight and ground software requirements
- Interface requirements documents
- System operation concept
- Trade studies
- Engineering analyses
- NEW! Software Assurance Classification Assessment & Report
  - New requirement defined by NPR 7150.2
  - Prepared by Project, independent assessment by Software Assurance

Flight Software Requirements Drivers

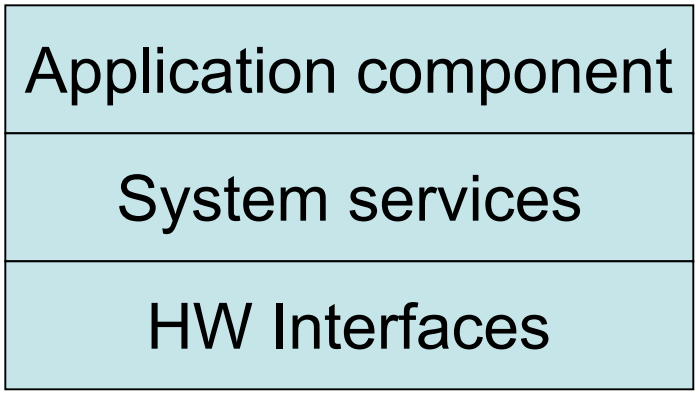


Simplified Flight SW Architecture



- Flight software applications communicate across bus
- Each application has similar architecture
  - Layering limits pervasiveness of change

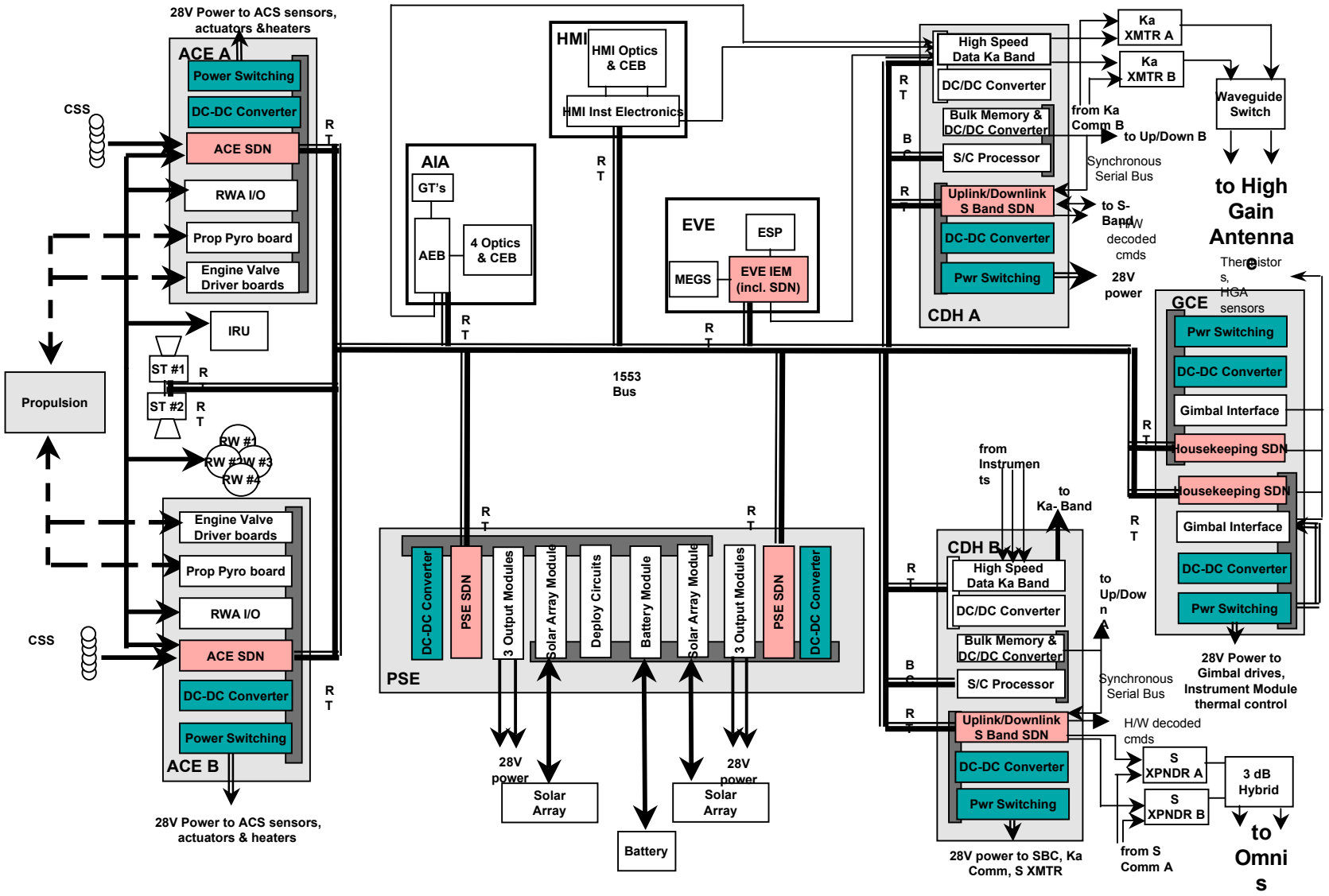
Application Architecture



What's Complex About Flight Software?

- Schedule Compression
  - Requirements & Operations Concepts mature late
    - Flight SW is **more tightly coupled to hardware** than ground SW
    - Can't finalize requirements until flight hardware is fully characterized
  - Major Demands during I&T and Launch Preparations
- Software's flexibility is both a blessing and a curse
  - FSW can often **accommodate** late requirements changes or compensate for hardware problems during I & T or on-orbit **if and only if one plans** the sufficient funds for testbeds, well defined regression tests, and staff
    - Under full cost accounting, contingency funds must be **planned** at the beginning of the project; changes are not "free" anymore.

SDO Electrical Architecture

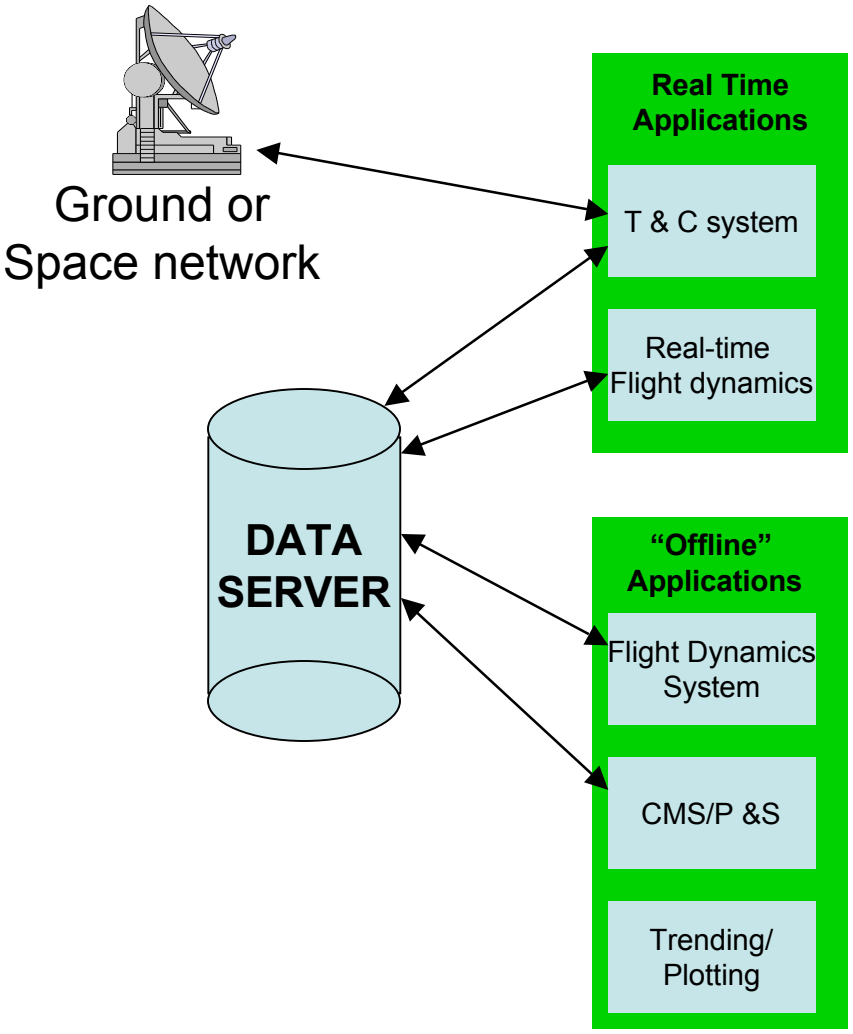


Multiple processors, buses and interfaces

Ground Software Requirements Drivers: Mission Operations Requirements

- Ground software is integral to **testing** as well as **operations**
- Spacecraft complexity
  - "Observatory class" spacecraft with multiple instruments, and large telemetry/command databases, usually require more complex ground systems than "explorer" class spacecraft.
  - Highly constrained flight data system resources (CPU, memory) increases ground system complexity, especially in the area of planning & scheduling
  - Spacecraft with lower degrees of redundancy and autonomy usually require more complex ground systems.

Mission Control Ground System: Simplified MOC Architecture

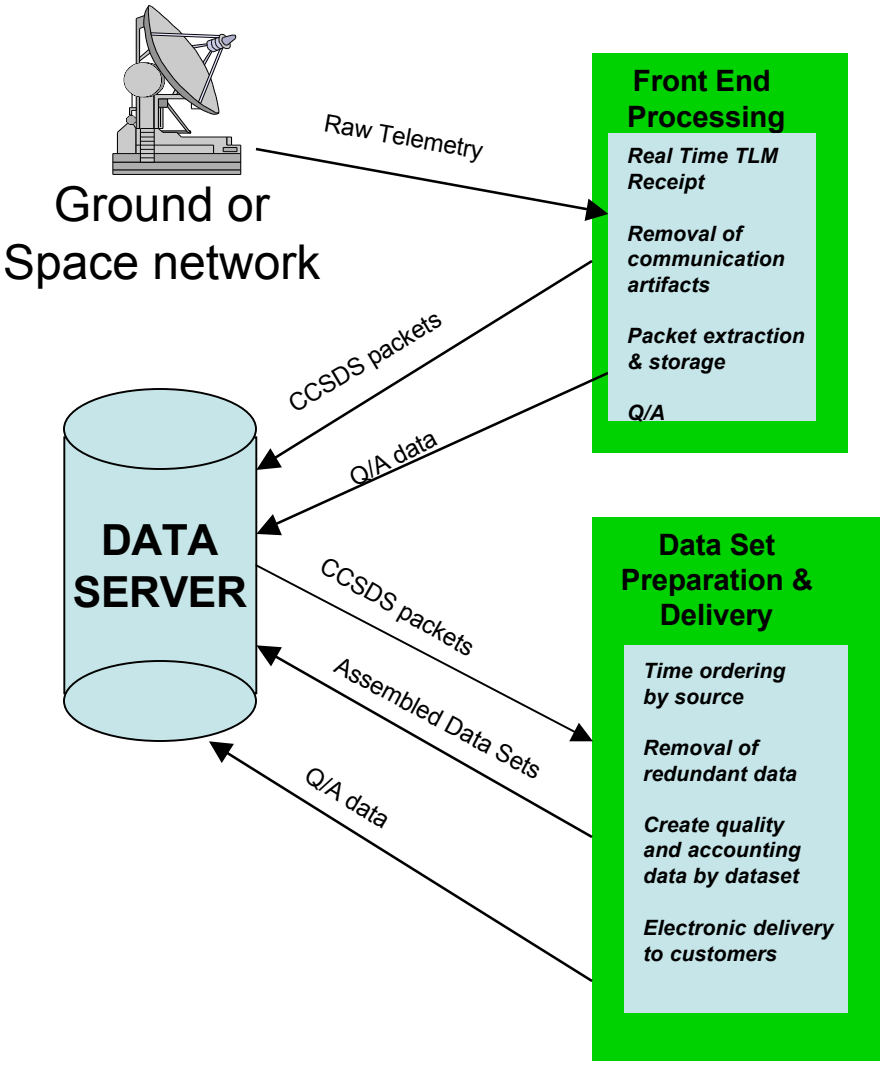


- Main MOC applications
  - Telemetry & Command system
  - Flight Dynamics
  - Command Management, including Planning & Scheduling
  - Trending
- Level 0 processing (LZP) of science data is moving to the MOC

Ground Software Requirements Drivers: Science Data Processing Requirements

- Mission Science Requirements/Instrument observation requirements
  - Instruments with targeted observation needs, tight pointing requirements introduce complexities into planning/scheduling system.
- Data rates
  - Very high (10-100's Mbps) average and peak downlink rates require custom hardware/software systems to support
- Data Latency and completeness requirements
  - In general, as data latency decreases, and data completeness requirements increase, ground system complexity increases

Science Data Processing: Level Zero Processing



- Level zero processing creates data sets of CCSDS packets, separated by data source, from raw downlink
- CCSDS standards minimize need for mission-specific SW
- HW architecture needed for LZP dependent upon mission data rate
  - Smaller missions (< 1 Mbps on average) can perform front end processing and data set preparation using workstations
  - Larger missions (e.g, EOS, SDO) require special purpose VLSI equipment and dedicated high-speed computers

Developing Requirements in GSFC Science Environment

- Collection of independent researchers who explore a wide variety of scientific disciplines
  - Generally small focused teams; limited resources
  - Requirements manager is part-time role of another full-time position
- Most requirements are gleaned in informal settings, particularly through workshops and individual discussions
- No “one size fits all”
  - Requirements engineering and development methodologies for one project will be very different on another
- Most projects have some way to adjust for change
- Each project has some form of documentation; not all projects have ways of tracing requirements
- COTS tools are desirable, but not affordable

What's Complex About Ground Systems? Mission Control Ground Systems

- Real-time ground system components often have high reliability, 1 minute restoral-of-service requirements
  - May not be running continuously, but needs to be reliable during contact time.
  - Significant test time needed to ensure requirements satisfied
- Must be highly automated/support unattended operations
  - Required to ensure reasonable operations costs for missions
- Application & operations complexity
  - E.g. flight dynamics for constellation missions
- Changes late in lifecycle (I & T or operations) are quite common
  - Usually cheaper to change & test ground than flight system.

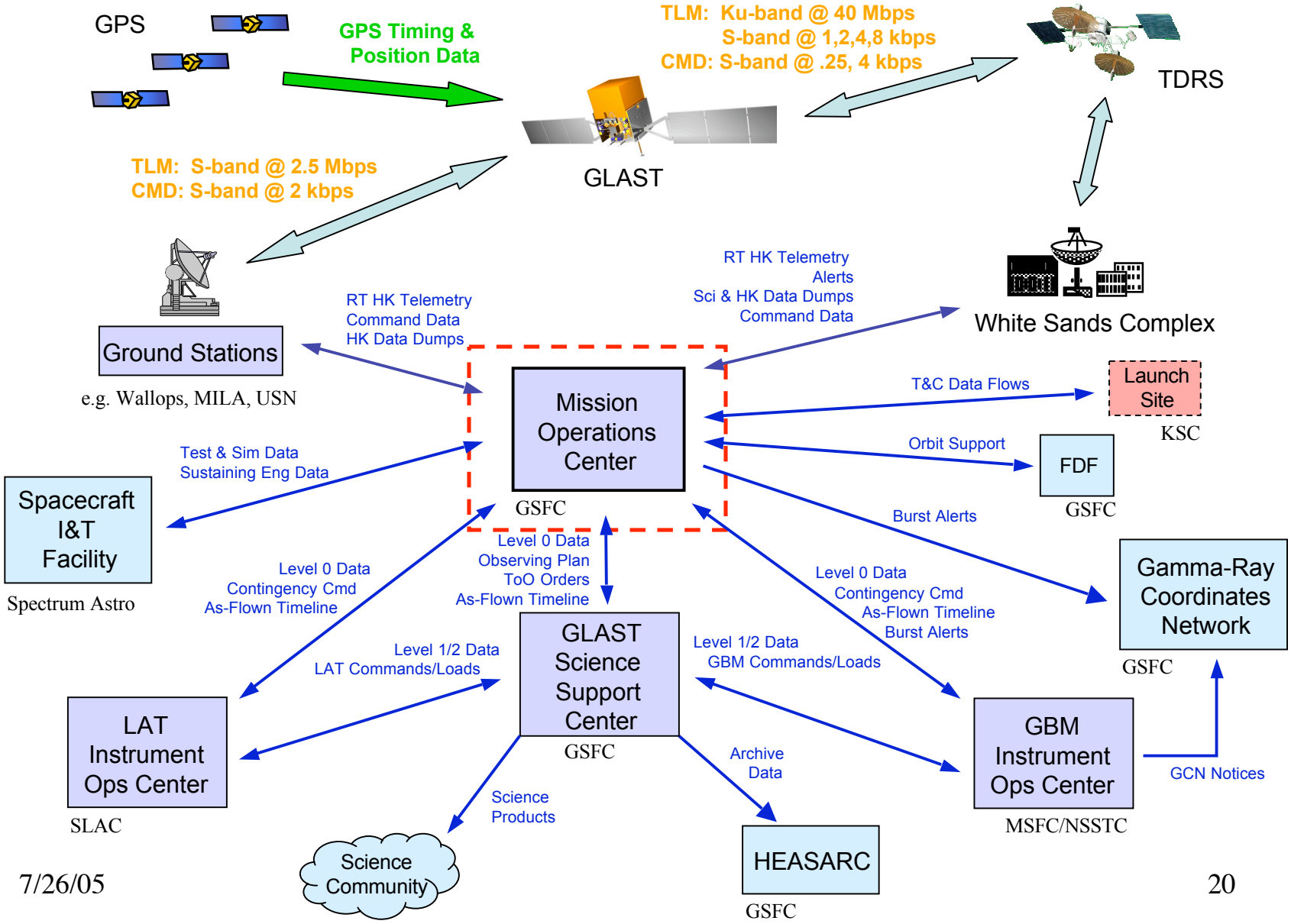
What's Complex About Ground Systems? Level Zero Science Data Processing (LZP)

- High data rates (10s – 100s Mbps) and low delivery latency (< 2-3 hours) requirements drive architectural complexity
  - Special purpose hardware/multi-CPU platforms needed
  - Reduction in ability to re-use previously developed software
- Number of data sets created per day
  - Function of the number of sensors on-board spacecraft, and timespan per data set
  - As the number of data sets created increases, quality/accounting and data set management/delivery software complexity increases
- "Compromises" to accepted telemetry standards (CCSDS)
  - Mission-unique software development required if CCSDS standards are not rigidly adhered to in flight system.

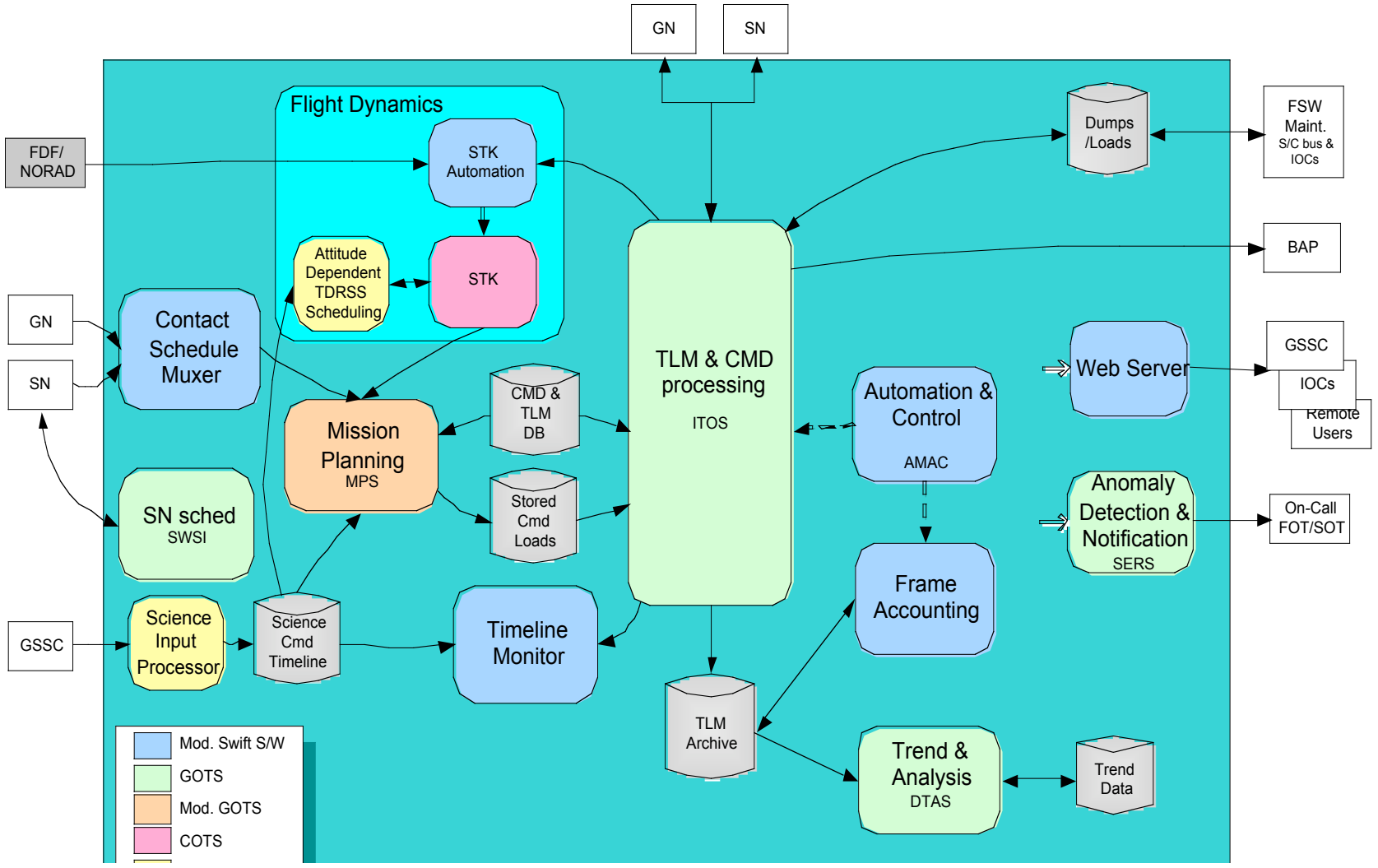
What's Complex About Ground Systems? Science Data Processing Beyond LZP

- Issues in further capturing, processing and archiving science data, esp. at instrument level
  - Numerous components & interfaces involved - many stages of processing
  - Components may involve execution of scientific codes at multiple sites using heterogeneous hardware platforms & operating systems
  - Each stage may involve different teams of developers
  - Very different quality standards may exist -- many science data processing codes developed by the scientists themselves
  - Factors to consider: Real time data needs, Science data parameters, Metadata standards, Science data format, instrument calibration, instrument specific processing
- Key to success is having robust interfaces between components
  - Interface control documents are critical
  - Must coordinate activities among teams

Ground System Architecture: GLAST



Mission Control Ground System: GLAST MOC Architecture



Extensive COTS & GOTS usage in MOC

"Acquisition" is Used in a Variety of Ways

- Rapid Spacecraft Development Office (RSDO) missions
  - Key issue is whether catalog components cover requirements
- Contracted development
  - SW as an independent contract or as part of system contract
    - Need sufficient deliverables to manage project
    - Can be some or all of the different types of software for a mission
    - "COTS" often used to describe this type project; but contracted development includes the services needed to configure COTS
- Purchase of COTS software for use on in-house software development
  - Different activities when developing software using COTS
  - Maintenance and upgrade issues

Best Practice: Designate a Software Manager to assess the Project's scope and to tailor the acquisition requirements

Cost Estimation:
Rules For Ballpark Development Cost Estimation

- Provided by experienced software managers
  - Some examples
    - Spacecraft flight SW: 50-80 staff years (MAP was 75)
    - Mission Control Ground SW: Great Observatory 60-80 staff years, Small Explorer 25-35 staff years
- General approach
  - Comparison with previous missions
  - Adjust based on technical & management risk factors
  - Example: Flight Software Branch feature-by-feature complexity estimation
    - Experienced developers rate complexity of each feature using previous missions as guide
    - Rationale for ratings should be documented
    - Scores assigned to complexity correlate with effort

Cost Estimation: Causes of Flight SW Cost Growth Explained

- Planning
  - Specific Project pressures to reduce cost estimates due to cost constraints, *in some instances never allowing a true estimate to be baselined*
  - Post-PDR mandatory cost reductions across the board
  - Unrealistic dependencies within delivery schedule (e.g., inputs to reqts. not provided as scheduled, hardware not available as scheduled)
  - Planning to reduce perceived overhead doesn't pay off
    - Minimized production or review of documentation
    - Minimized formal and informal review activities
- Requirements
  - New requirements often levied in late mission design or I&T phase, as systems personnel become more knowledgeable
  - Incomplete requirements
  - Development and test ramifications are not fully understood

Cost Estimation: Causes of Ground SW Cost Growth Explained

- Planning
  - Project pressure to reduce overall cost estimates
  - COTS capabilities not functioning as planned/advertised
- Requirements
  - Maturity of requirements limits accuracy of preliminary cost estimates
  - Disparity in completeness of functional requirements (e.g., telemetry and command requirements are often well understood, while science processing requirements are left incomplete until late in mission lifecycle)
- Testing
  - Insufficient emphasis on test plan completeness
  - Application of additional testing as primary risk mitigation
- Staffing
  - Launch delays result in funding and maintaining "marching army"
  - Lack of adequate dedicated staffing early in project lifecycle

Software Related Trades

- Selection of Ground System (telemetry & command)
  - Issue: GSFC GOTS are designed with flight SW test in mind, many COTS ground systems are not
- Hardware architecture of flight data system
  - Issues: sufficient hardware capacity, complexity of testing distributed systems, capabilities of operating systems, compilers and related tools
- Sufficient testbed resources
  - Issue: Relying on flight hardware as a testbed is not sufficient. What is needed is a full ETU version of all Flight Data System components (high fidelity testbed).
- Allocation of functionality between flight & ground systems
  - Issue: Many operations activities can be performed either by flight or ground segment. Assigning functionality to flight software in a prudent manner will decrease complexity of ground software development and operations
- Interface compatibility between ground system components
  - Issue: Standard interface methods (data formats, frequency of message exchange, etc.) do not currently exist between key ground system components.
- Degree of onboard science data processing
  - Issue: May be able to exploit more capable flight hardware and operating systems

Formulation Phase: Summary of Best Software Practices

- Software team needs to be fully integrated into the formulation process
- Plan for sufficient development tools, testbeds & simulators
- Plan cost growth of 20% from PDR to launch readiness
- Question reuse assumptions, evaluate COTS software carefully
- Make sure RFP is complete with respect to deliverables, metrics, and development processes
- Start software development with attention to
  - Organizational structure
  - Software management plan / product plan

Agenda: Software Development Phase

- Key Documents and Deliverables
- Software Development Considerations
  - Software Development Life Cycle
  - Reviews, Inspections & Walkthroughs
  - Testing Considerations
  - Assurance Considerations
- Managing Software Development
  - Project Planning
  - Risk Management
  - Project Monitoring and Control
  - Post-development support
- Technology Development
- Improvement Initiatives

Key Documents / Deliverables (More Than Code)

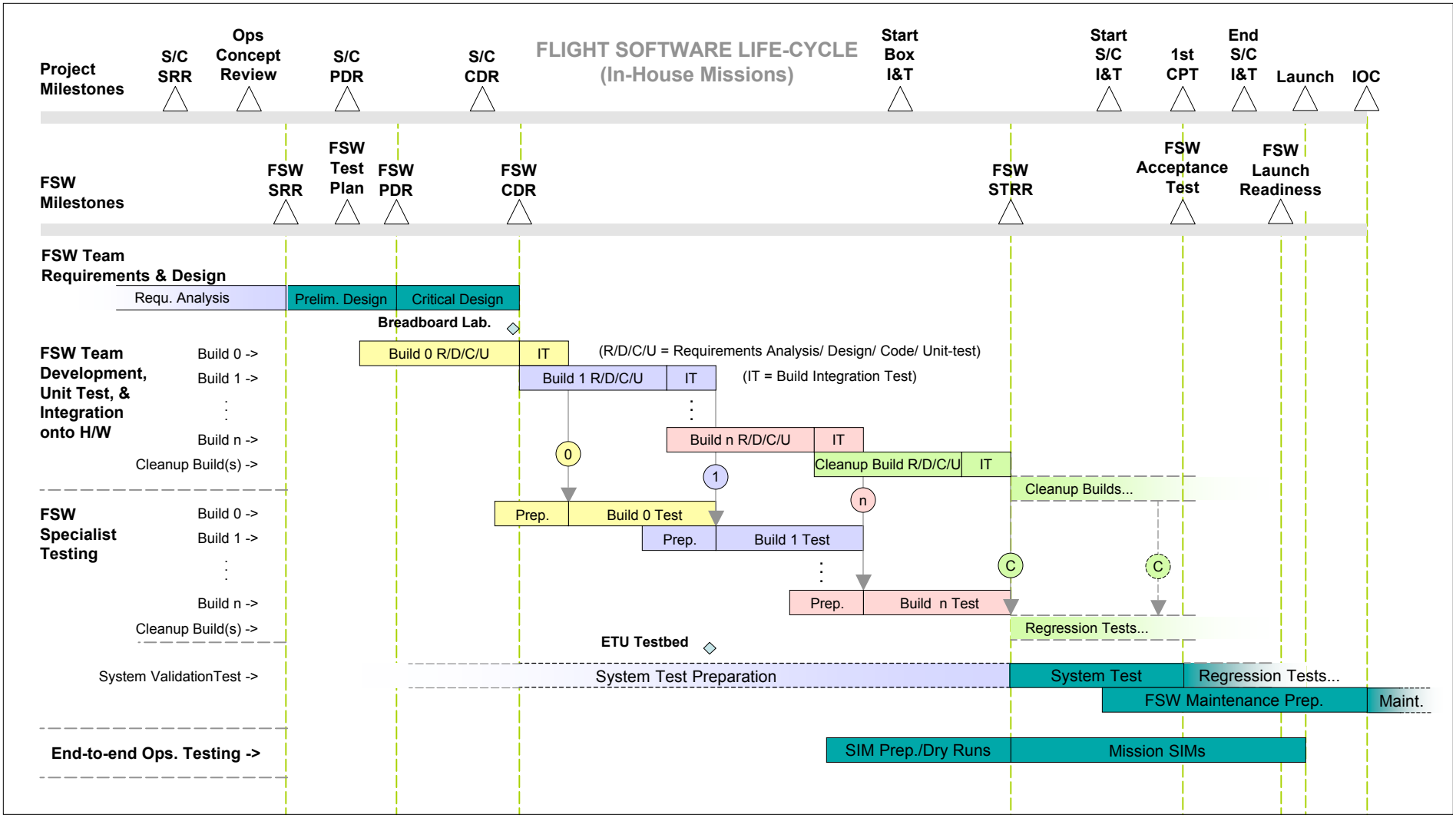
- Software life-cycle products include
  - **Product Plan / Software Management Plan (may include Configuration Management Plan, Risk Management Plan, Security Plan, or they can be separate documents)**
  - **Software Requirements Document (SRD)**
  - **Interface Requirement Document(s) (IRDs) & Interface Control Documents (ICDs)**
  - **Software Quality Assurance Plan**
  - Software Requirements traceability matrix
  - Functional and Detailed Design Documents
  - ***Source code***
  - **Software Test Plan(s)**
  - Test and Verification Matrix
  - Software User's Guide
  - Release Letters
  - Delivery, Installation, Operations, and Maintenance Plan(s)
  - Software Safety Plan (can be part of System Safety Plan or separate)
  - Derived planning information (includes build plans allocating functions to build, progress tracking spreadsheets)

Boldface indicates CCB control

Software Development Life Cycle: Description

- Most projects use iterative life-cycles, but all show some form of "requirements-design-build-test."
- Most life-cycles are depicted as GANTT-type schedules.
- Software life cycles have intermediate milestone reviews and deliverable products that provide completion and coordination points with other project elements or other sub-processes
  - **Build**: A version of a system or component that incorporates a specified subset of the capabilities that the final product will provide.
  - **Release**: a version that is made available for use outside the development team
- Alternate life-cycle models include prototyping, spiral modals, and agile methods (such as XP). These models are useful when used correctly, but **they can be misused to circumvent sound software development practices**; SW lead can advise on their use

Software Development Life Cycle: Software Life Cycle in Project Context



Flight SW example, ground is similar

Software Development Life Cycle: Software Build Planning Considerations

- Integrate early, integrate often
  - Plan builds at 3-6 month intervals
  - Testable increments of functionality
  - Early detection of interface issues
  - Early validation of development and test practices
- Consider hardware & software test schedule dependencies
  - Ground system & testbed delivery schedule for flight SW
  - Meet schedule to support flight data system box testing
- Consider risk
  - Put higher complexity software in early builds
  - Defer incomplete or unstable requirements to late builds
  - Have early flight SW build with full closed-loop operations
  - Focus on interfaces *between* COTS or GOTS products

Reviews, Inspections & Walkthroughs: Milestone Reviews

- Purpose
  - Baseline key work products
  - Present budget, schedule & risk status
    - **Estimates should be updated based on available information**
    - **Expect cost growth of 20% between PDR and launch readiness**
- Hold software review **after** corresponding system level review (SRR,PDR,CDR)
  - Flight SW depends on system & HW architecture decisions
  - Ground SW depends on command & telemetry definitions
- Preparing for review helps
  - **Team understanding** of the product being developed
  - **Communications** within team and with other groups

Reviews, Inspections & Walkthroughs: Inspections & Walkthrough Processes

- Inspections are concerned with discovering defects; Walkthroughs are concerned with team reaching common understanding of product (e.g. requirements)
- Participants are: Author of requirements, design, code, test plan, Peer reviewers from development or test team, collaborators from relevant disciplines, quality assurance personnel
- A study of inspections at NASA (2001) interviewed project personnel and found: Improved communication between developers, training benefit, defect detection
- Early defect detection yields cost savings (JPL data)
  - Inspections cost 0.7 hrs / defect,
    testing ranges 5 to 18 hours / defect
  - Inspections caught over 70 per cent of defects

Software Inspections & Walkthroughs meet GSFC Engineering Peer Review Requirements (GPR8700.6)

Testing Considerations: Flight Software Test Activities

- During Software Development
  - **Developers** perform unit and build integration tests
  - **Test team** performs requirements driven build verification tests, and operational scenario driven system validation tests.
  - Acceptance test is last run of full system test suite.
    - Flight SW is ready to run in operational environment (on spacecraft)
    - Complete by the first Comprehensive Performance Test (CPT)
- After Software Release
  - Make changes in response to non-conformance reports
  - Run regression tests
    - Regression tests involve unit, build, and system test activities
    - Test change, subset of previous tests to uncover new errors in SW
- Need **breadboards** for integration of first build; need **Engineering Test Units (ETUs)** for subsequent builds

Testing Considerations: Ground Software Testing

- Test environment
  - Key interfaces need to be tested early and thoroughly
    - Is correct data being passed between facilities?
    - Is the system performance meeting requirements?
  - Target computer is often same as development system
    - Subsystems can often be tested thoroughly on development machines
    - "Soft real time" performance requirements
  - Acceptance tests still must reflect operational environment
- Flight operations team and science teams operating instruments are key to successful testing & deployment of ground system
  - Start FOT training early as initial builds are delivered
  - Need operator feedback on procedures, user interfaces

Assurance Considerations: Comparing Software Quality and IV&V

Software Quality

- Provides **Center-level** services
- Focuses on **ALL** Project software
- **Emphasizes compliance to standards and procedures**
- Reviews, monitors and audits all Project processes and products for completeness and accuracy
- Matrixed to the Project as part of the Project Team and provides daily insight/oversight
- Reports to Project and Center Director through OSSMA

IV&V

- Provides **Agency-level** services
- Focuses on **MISSION CRITICAL** Project software
- **Emphasizes completeness and correctness of the product**
- Reviews, analyzes, and provides in-depth evaluations of life cycle products which have the highest risk
- Independent from the Project and provides analyses and evaluations per IV&V priorities
- Reports to Project, GPMC, and NASA Headquarters

Assurance Considerations: Software Safety

- **Software Safety**
  - Entails identifying, analyzing, tracking, mitigating and controlling **software hazards and hazardous functions** (data and commands) to ensure safer software operation within a system
  - Done as part of **system** safety analysis
  - Software is safety critical even if there is hardware backup
  - Adheres to the requirements specified in the NASA Software Safety Standard, NASA-STD-8719.13B
  - Safety "Litmus Test", found in 8719.13B, will help in determining the criticality of the software and its contribution to the safety of the system

Software Safety is a function of System Safety!

Project Planning:
Software Product Plan

- All code 580 in-house software development activities shall have a Software Product Plan
  - Product Plan may serve as SW Management Plan
- ISO 9000 requirement – equivalent of Hardware Work Authorization (WOA)
  - Outline for the plan is on the web at http://isd.gsfc.nasa.gov/iso9k/iso9001.htm
  - Major areas of the plan:
    - **Customer Agreement**
      - Overview of goals & objectives for software product
      - Source documents for software requirements
      - **Resources required (CS labor, contractor labor, other costs)**
      - **Receivables & deliverables**
      - **Software team placement & reporting relationships**
      - **Acceptance criteria (pointers to acceptance test plans)**
      - Customer training
      - Post delivery maintenance
    - Management Approach
    - Technical Approach
    - Product Assurance

Risk Management

- Inherent uncertainty in estimation methods can be handled by producing worst, most likely and best case estimates
- "I forgots" can be handled using standard processes and products (standard WBS, checklists, document templates, cost estimation procedures,…)
  - Example: Is SQA and IV & V in budget?
- "Unknown unknowns" can be handled by keeping a management reserve
  - Example: late SW requirements change as HW workaround
- Look at what problems have occurred in the past
  - For example, Flight Software Branch risk tool includes a list of "generic risks" based on previous projects; flight software lead can select which ones apply
- "Known unknowns" can be handled by identifying, analyzing and mitigating risks. Examples:
  - Late and changing requirements
  - Compressed schedule
  - Late hardware deliveries / untested hardware
  - Inadequate test beds
  - Inadequate COTS -- buggy, doesn't cover requirements
  - New technology

Project Monitoring and Control: Is There Enough Information to Judge?

- Are metrics vague and high level, or are objective measurements being used?
  - "we are 90% done"
  - ***Good metrics provide objective data to judge progress***
- Do status reports always sounds the same?
  - "we are 90% done"
  - ***There are specific alarm signs one should look for***
    - Requirements, management, process, schedule

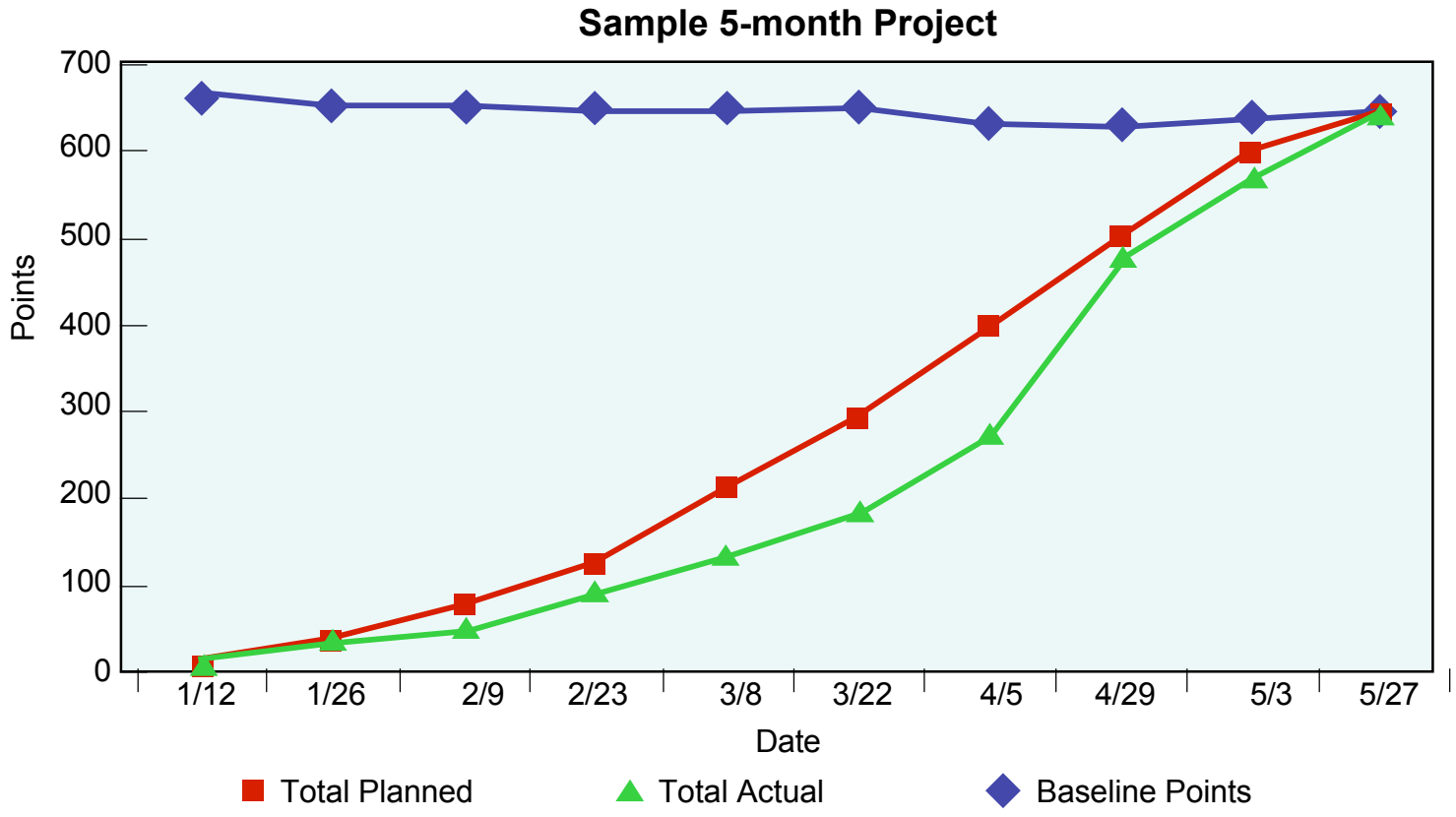
The Big Question

"***Why do you think you will be able to deliver on time?*** "

Project Monitoring & Control: Required Metrics

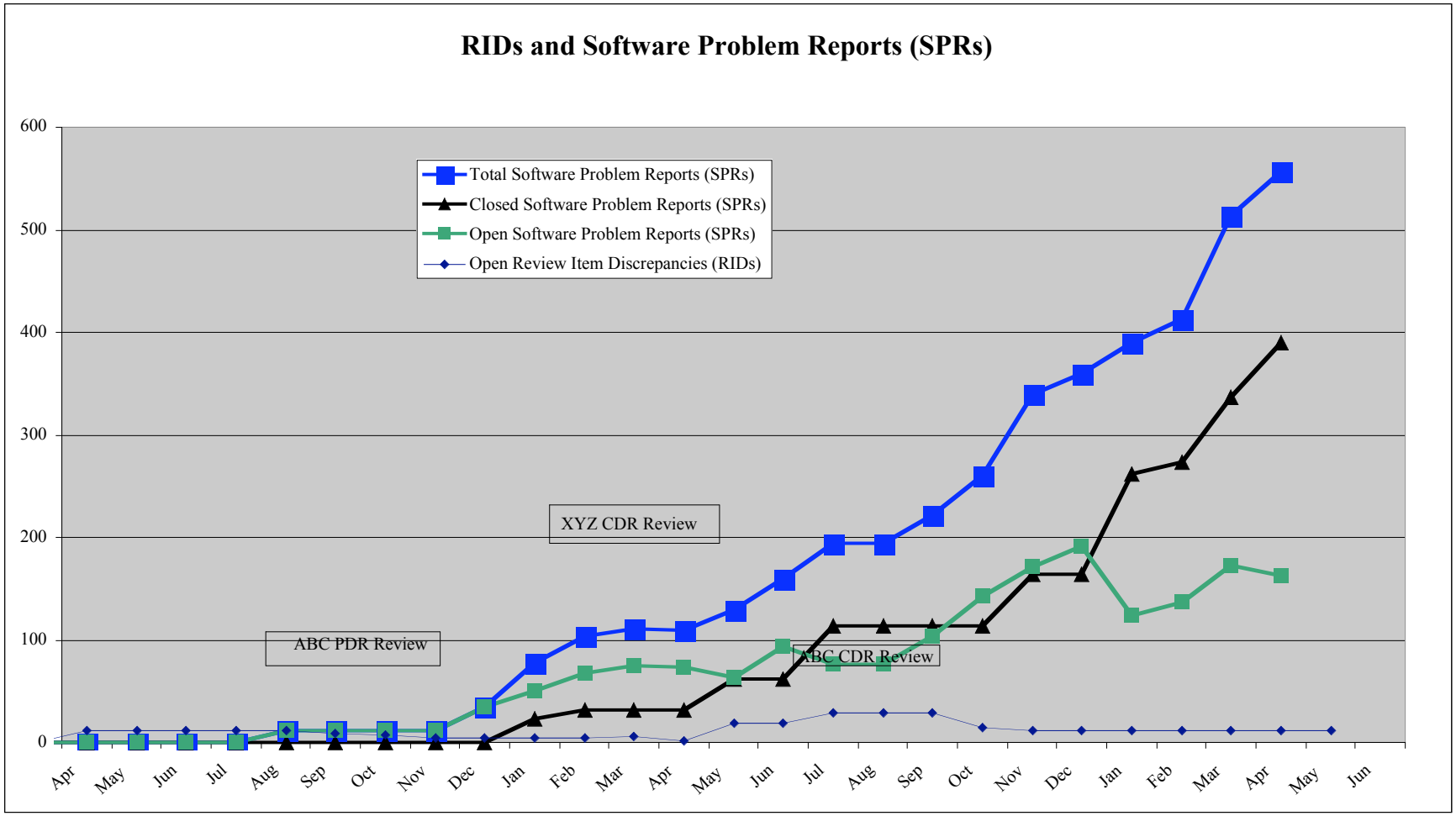
- Progress tracking data
  - Includes cost, staff effort & scheduled completion of work
  - Track planned progress against actual performance
- Defect data
  - Open, closed, and total number of discrepancy reports
  - Number of defects found in inspections or walkthroughs
  - Effort spent in inspection, walkthrough & testing activity
- Number and impact of requirements changes
- Current & projected use of system resources
  - Main memory, secondary storage, bus & CPU cycles,…

Project Monitoring & Control: Progress Tracking for Subsystem Development



Each module (165) assigned 4 points: 1-designed; 2- coded; 3-inspected; 4-integrated

Project Monitoring & Control : Defect Reports and Product Quality



Project Monitoring & Control : Alarms to Monitor

- Ask questions related to potential problems
  - Requirements, management, process or schedule
  - Each alarm has associated corrective actions
- **Consequences of ignoring alarms**:
  - **Quality is sacrificed**
    - "we'll cut testing to get back on schedule"
    - Cut out walkthroughs & inspections to keep earned value on track
      - Progress should be recorded only on completion of steps
  - **Work is deferred, costs soar**
    - Moving to later builds will require resources not in original plan
  - **Major staff increases to meet schedule**

Project Monitoring & Control: Alarms

- Is there a mismatch between requirements and budget?
- Are the requirements mature and stable?
- Do you have the right personnel?
- Is there sufficient communication with software teams?
- Are processes being executed correctly or evaded?
- Is the development team suffering from schedule compression?
  - Are builds being combined to “reduce overhead”? Are capabilities being moved to later builds? Are early build mature enough to support implementation of later ones?
  - Are hardware and algorithms needed by software team delivered on time?
  - Do developers have time for all activities, including support for other groups?
  - Is planned documentation being produced?
- Is the test team suffering from schedule compression?
  - Are testbeds delivered to meet the test schedule?
  - Are multiple builds being tested simultaneously?
  - Are planned tests being cut out to match resources?
  - Are the test procedures and expected results stable?
  - Does test team have enough time to analyze results?

Project Management Considerations: Summary Recommendations for Development

- Recognize that there is more to software than code
- Monitor the same items for contracted and in house development
- Allow time in schedule for reviews, inspections & walkthroughs
  - Problems detected early are less costly to fix
  - "Testing in quality" is far more expensive; not an appropriate approach to software development
- Have frequent, testable builds with incremental functionality
  - Help find interface / integration problems early
- Keep estimates up to date
  - Cost & schedule
  - Underlying assumptions

Post-Development Support: Post-Acceptance Deliverables and Activities

- Once software is accepted and delivered, discrepancies and changes shall be tracked using NCRs.
- Support of delivered software through in-orbit checkout
  - Spacecraft Integration & Test
  - Mission Simulations
  - Launch & early orbit checkout
- Software maintenance (sustaining engineering post-IOC)
  - Identify, prioritize, design, implement & test changes
  - Regression testing

Acceptance Tested Software is **ready to run in operational environment**

Post-Development Support: Key Recommendations

- Schedules and dependencies
  - Flight SW acceptance test should be complete before first spacecraft Comprehensive Performance Test
  - Ground elements should be ready in a similar time frame
- Plan late-phase resources at the start of the project
  - Software team support for flight operation, hardware I & T, and observatory/payload I & T teams.
  - On-orbit sustaining engineering
  - Licenses for and upgrades to COTS products
  - All software will need maintenance.
    - Maintenance must be included in planning.
    - What and how much depends on the use of the software.

ISD Technology Research & Development

- Goal is to infuse new technologies and/or new approaches into mission control and science systems.
- Ideas are generated through
  - Strategic technology planning
  - Meeting with Goddard missions and scientists
  - Review of NASA Strategic Plans
- Potential technologies are found via literature research, conferences, educational opportunities, and market awareness.
- Most efforts are funded on a yearly basis. The goal is that at the end of each funding year a useful prototype will be completed.
- Teams are small, normally 1 to 3 people.

ISD Technology R & D Process

- Review technology proposal calls and develop proposals to respond to those calls
  - Calls are both internal and external to Goddard
  - Submitted proposals normally have a potential customer associated with the concept who has agreed to beta test the developed software
- Upon proposal selection develop project plan in accordance with funding source requirements.
- Development plan uses the iterative prototyping approach
  - Normally projects have a quarterly builds that are reviewed by customer advocate.
  - Customer feedback is fed into the design phase for the next build.
- Projects have bi-monthly meetings with branch management to ensure that they are staying aligned with NASA strategic objectives.
- Most proposal funding sources require yearly project reporting.

ISD Technology Projects

- **Adaptive Sensor Fleet** (ASF) is a supervisory control system that is designed to use a collection of heterogeneous robotic platforms to optimally perform observations of dynamic environments driven by high-level goals.
- **Instrument Remote Control** (IRC) is a cross-platform, distributed control framework for instrument that provides remote configuration, control, monitoring, and analysis
- **Multipurpose Exoterrain for Robotic Studies** (MERS) is a facility available to test robots, sensors and advanced exploration concepts in a semi-realistic environment
- **Science Algorithm Visualization and Networking Tool** (SAVANT) is a graphical tool for creating "visual programs" by assembling and executing flowcharts (graphs) of web services.
- **Science Goal Monitor** (SGM) is a tool used to improve both the science data acquisition and data quality of a mission by automatically responds to science-related events.
- **Virtual Feel** is a prototype tool set that will accomplish the typical assembly/disassembly tasks needed for the Hubble Robotic Repair mission through a new approach which will significantly reduce the risk of failure.

Improvement Initiatives

- Process Initiative
  - GSFC Software Process Improvement Project
    - Point of contact: Sally Godfrey
    - http://software.gsfc.nasa.gov
- Technology Initiatives
  - Flight Software Reuse
    - Point of contact: Jane Marquart
  - GSFC Mission Services Evolution Center (GMSEC)
    - Point of contact: Dan Smith
    - gmsec.gsfc.nasa.gov
- What's in it for Projects?
  - ***Projects can leverage these improvement investments for better, more cost effective systems.***

Improvement Initiatives: Software Process Improvement

- Support of NASA Software Improvement Initiative
  - GSFC response is the SPI Project, carried out by the Engineering Process Group (EPG)
  - ISD processes are the initial focus, as the division is responsible for the majority of GSFC mission software
    - ISD personnel lead the SPI Project and the EPG.
- GSFC SPI phases
  - Phase 1 (FY02) -- pilot appraisals, EPG startup
  - Phase 2 (FY03-07) -- staged process improvement
    - All ISD mission software will be at CMMI Maturity Level 3 by the end of FY07 (i.e. Software projects rely on **organizational process assets** to carry out their work, achieving more consistent performance)
  - Phase 3 (FY08 onward) -- sustain continuous improvement

ISD Initiatives: Flight Software Reuse

- Object Model of Mission-independent Core FSW Elements
  - Multi-mission FSW Requirements Analysis
  - Separate mission-unique and common FSW Requirements
  - Baseline Generic FSW Executive Architecture Definition
- Develop Common Architecture & Development Tool Set
  - Make the tools satisfy FSW needs
  - Improve FSW Documentation
- Validate, Measure Performance, Evaluate and Refine Re-use
- Train for re-use

Planned instead of ad hoc reuse

ISD Initiatives:
GSFC Mission Services Evolution Center (GMSEC)

•Introduction

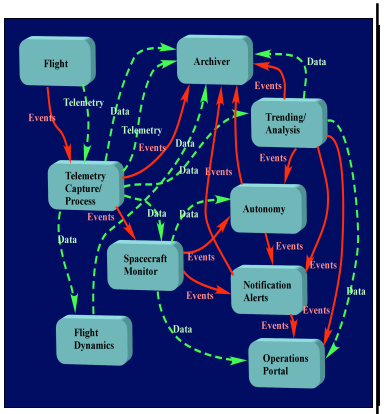
GMSEC was established in 2001 to coordinate ground and flight data systems development and services at GSFC.

•Goals

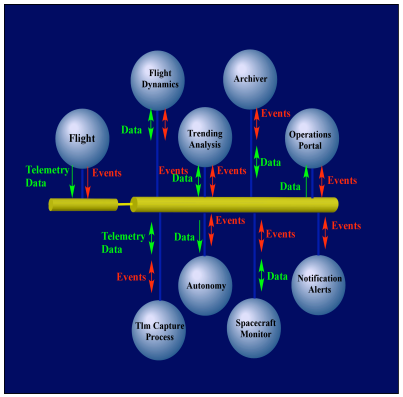
1. To simplify initial integration/development
2. To facilitate technology infusion over time
3. To allow for additional operations concepts
4. To establish a structure which can grow to enable future GSFC missions

•Concepts

1. Standardize Interfaces – not Components
2. Middleware infrastructure w/ publish/subscribe
3. User Choices – GMSEC doesn't decide which products are best
4. GMSEC "Owns" the Architecture and Interfaces – NASA AETD Branches still own their domain areas



Traditional "socket-based" system designs require complex interfaces and integration.



The GMSEC "Information Bus" approach has allowed some components to integrate on the first try.

Status

- Architecture developed in FY02; Lab established and demonstrations held in FY03
- COTS industry has helped develop message standards, see great advantages to GMSEC
- NASA ARC, MSFC, JPL and APL are interested in collaborating with GMSEC approach

Summary: To Strike Fear in the Heart …

- The often cited Standish Group 1998 Study on large software projects in the late 1990s reported:
  - 53% either delivered late or exceeded budget
  - 31% were cancelled
  - 16% were successful
- Failed projects average
  - 189% of the original estimated cost
  - 222% of original schedule
  - 61% of original functionality
- **Only 41% of IT managers believe there are fewer failures now than five years previously!**

Goddard is not exempt from software problems…

- EOS Flight Operations Segment (FOS) control center architecture
  - Reuse and COTS less effective than expected
  - Schedule metrics and requirements poorly defined
  - **Consequence**: **Terra launch delayed a year**
- Flight systems: IRAC instrument software, EO-1 Command & Data Handling Software
  - Poor control over requirements
  - Reliance on flight hardware as high fidelity testbed
  - Late involvement of Flight Software Branch
  - **Consequence**: **65% cost growth on EO-1, 245% on IRAC**

…but we generally do well, and we want to do better still

- Over the 5 years to July 2002 GSFC held responsibility for well over 25 missions or well over 125 mission critical SW elements
- GSFC has experienced 3 significant problems, computing to less than a 2.5% significant problem rate:
  - EOS FOS, EO-1 FSW, and IRAC FSW
  - Each was significant and drew a lot of attention
- No GSFC software problem has directly contributed to in-flight damage; to the contrary, software is routinely used to compensate for problems on-orbit
- *To further improve performance and to reduce hero mode dependence, pragmatic improvement steps can be taken*

If the process works…improve it

ISD Recommendations for Mission Success

- **Involve software personnel from the beginning**
  - Software experts' inputs improve cost estimates, trade studies, and acquisition plans
- **Produce a Software Management Plan / Product Plan (SMP/PP)**
  - Make sure "Customer Agreement" is acceptable to all
- **Follow the processes defined for the Project in the SMP/PP**
  - Cutting corners only helps you lose your way
- **Track the software development against the plan**
  - Keep an eye on the potential alarm signs